(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 March 2001 (22.03.2001)

PCT

(10) International Publication Number
WO 01/20483 A2

(51) International Patent Classification⁷: G06F 17/00

(21) International Application Number: PCT/US00/25287

(22) International Filing Date: 14 September 2000 (14.09.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/153,768	14 September 1999 (14.09.1999)	US
09/556,086	21 April 2000 (21.04.2000)	US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier applications:

US	60/153,768 (CIP)
Filed on	14 September 1999 (14.09.1999)
US	09/556,086 (CIP)
Filed on	21 April 2000 (21.04.2000)

(71) Applicant *(for all designated States except US)*: CANTAMETRIX, INC. [US/US]; Suite 690, 110 -110th Avenue Northeast, Bellevue, WA 98004 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: WELLS, Maxwell, J. [GB/US]; 6817 - 44th Avenue Northeast, Seattle, WA 98115 (US). WALLER, David [US/US]; 7634 Hollister Avenue #351, Goleta, CA 93117 (US). DHILLON, Navdeep, S. [GB/US]; 8011 - 29th Avenue Northwest, Seattle, WA 98117 (US).

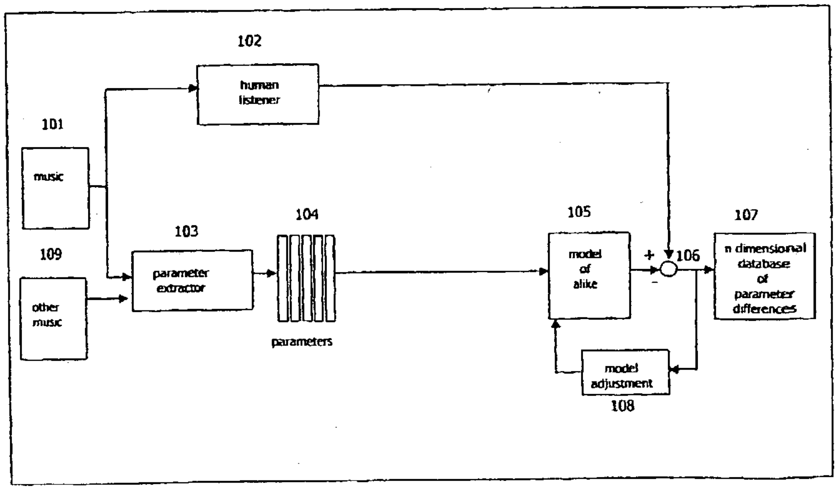
(74) Agents: HALEY, Jeffrey, T. et al.; Graybeal Jackson Haley LLP, Suite 350, 155 - 108th Avenue Northeast, Bellevue, WA 98004 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE,

[Continued on next page]

(54) Title: FINDING PERCEPTUALLY SIMILAR MUSIC WITH PRECOMPUTATION



(57) **Abstract:** A system is disclosed which allows people who enjoy listening to musical recordings to find new musical recordings that are similar target musical recording. The system uses a model of alikeness (105) which is based upon empirical observations of human perception of likeness between pairs of songs. The humans judge a representative sample of music (101) to build the model (105). Then other musical recordings (109) are applied to this system which extracts parameters (104) from the content of each recording and applies them to the model (105) to compute the differences between the parameters of the new recording and parameters developed for the model to place them in a data base of parameter differences (107). By allowing a user to select one of the recordings from the data base for which parameters have already been extracted, the system can efficiently work with a large number of recordings. A further efficiency improvement is achieved by precomputing differences between the target recording and other typical recordings within the data base. A large data base is efficiently managed by partitioning the data base into clusters.

IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

FINDING PERCEPTUALLY SIMILAR MUSIC WITH PRECOMPUTATION

1.0 BACKGROUND

Modern computers have made possible the efficient assemblage and searching of large databases of information. Text-based information can be searched for key words. Until recently, databases containing recordings of music could only be searched via the textual metadata associated with each recording rather than via the acoustical content of the music itself. The metadata includes information such as title, artist, duration, publisher, classification applied by publisher or others ("genre"), instrumentation, and recording methods. For several reasons, it is highly desirable to be able to search the content of the music to find music which sounds to humans like other music. One reason is that searching by sound requires less knowledge on the part of the searcher; they don't have to know, for example, the names of artists or titles; all they have to know is that they like a particular song and want to find another that is similar. A second reason is that textual metadata tends to put music into classes or genres, and a search in one genre can limit the discovery of songs from other genres that may be attractive to a listener. Often the genre label is applied at the artist or album level, and this label may be inappropriate at the song level. Yet another reason is that searching by the content of the music allows searches when textual information is absent, inaccurate, or inconsistent.

A company called Muscle Fish LLC in Berkeley, California has developed computer methods for classification, search and retrieval of all kinds of sound recordings. These methods are based on computationally extracting many "parameters" from each sound recording to develop a vector, containing a large number of data points, which characteristically describes or represents the sound. These methods are described in a paper entitled Classification, Search, and Retrieval of Audio by *Erling Wold, Thom Blum, Douglas Keislar, and James Wheaton* which is was published in September 1999 on the Muscle Fish website at Musclefish.com, and in US Patent 5,918,223 to Blum et al entitled "Method and article of manufacture for content-based analysis, storage, retrieval, and segmentation of audio information."

In the Blum system, a large number of vectors, each representing a brief sound recording, is assembled in a data base. A user can specify ranges for each of the parameters to find within the data base a desired sound. Alternatively, the user can present to the system a new sound recording. The system then computes parameters for the recording, assembles the parameters into a vector, and then, by vector subtraction, computes the distance from the example sound recording to the previously stored sound recordings and presents to the user the sound recordings whose vectors are closest. Using this "query by example" method, a user can present a new sound to the data base and find other sounds within the data base that are similar.

2.0 SUMMARY OF THE INVENTION

The invention builds on the Blum method of extracting many parameters from each of many sound recordings and assembling them into a database. Like Blum, to build a database of sounds, a set of parameters is extracted from a large number of recordings. Vectors of the parameters (or "descriptors" derived from the parameters as specified in US patent application number 09/556,086 which is incorporated by reference) are assembled and compared by subtraction or by finding the correlation. The present invention adds improvements that are particularly adapted to music and allow humans to find desired musical recordings within a large database of musical recordings.

To perform "query by example", instead of receiving a new sample of sound and performing all of the necessary computations of parameters and vector differences, the system allows a user to select one of the sounds already within the data base as the query starting point. By starting with a sound already in the database, the parameters have already been computed, the vector has already been assembled, and the distances to other vectors have already been computed. This precomputation allows the improved system to operate much more quickly than the Blum system. With this improvement, the system becomes practical for helping users find, within a very large database of music, music which is similar to music they already know. Also, by using a recording for which the parameters have already been extracted, copies of the system can be distributed without including parameter extraction methods that should not be made available. Similar benefits are obtained if the system also includes precomputed vector comparisons so that vector difference computation methods can be kept confidential.

In one embodiment of the invention, a business entity performs computation of parameters for all musical recordings to which it can gain access and then widely distributes these parameters, with identifiers specifying the recordings, to large numbers of people and computer systems throughout the world. By precomputing and widely distributing the parameters, others throughout the world will have the parameters to work with and will not need to compute the parameters themselves, and it provides a standardized computation of parameters to make them comparable from one recording to another. Others can then assemble sets of musical recording identifiers for which the parameters are available and use these selected sets to specify a search domain for finding music that is similar to a target musical recording where the parameters were also precomuted by the same original process. This will allow the creativity of large numbers of people to be applied to selecting sets of musical recordings for search domains that can be made available to others for finding music similar to a target recording. Computation of the difference measures between the target and the search domain can be done on a user's computer or on a server across a network.

For a very large data base of sound recordings, large enough to store the large numbers of available music recordings, the challenge of pre-computing the differences between each pair of songs becomes formidable, even for the fastest computers. While the number of combinations of 10 songs taken two at a time (45) is only 4.5 times the number of songs, the number of combinations of 100 songs taken two at a time (4,950) is 49.5 times the number of songs, the number of combinations of 1000 songs taken two at a time (499,500) is 499 times the number of songs, and the number of combinations of 10,000 songs taken two at a time (49,995,000) is 4,999 times the number of songs. (The number of combinations increases as the square of the number of songs.) To make this pre-computation challenge manageable, the data base is organized into clusters where the songs are similar within a cluster and a first computation is made with one representative song from each cluster. In essence, the data base is "partitioned" into manageable sizes for computing differences between vectors.

As a further improvement, instead of simply computing simple vector differences, the computation is adjusted with weightings to reflect a model of how humans perceive likeness between pairs of songs. One or more humans listen to pairs of songs and judge their similarity on a scale, for example from 1 to 9, where 1 is very dissimilar, and 9 is very similar. The objective of using human listeners is to create a model of their perceptual process of "likeness", using a subset of all music, and to apply that model to the set of all music. To build the model, a list of numbers is calculated for the comparison of each song to each other song. The model then yields a number that predicts the likeness that people would judge for the same two songs. The model processes the list of difference values by applying weights and heuristics to each value.

In yet another improvement, the results of searches of the database are filtered to improve the acceptance of the returned matches. Very large databases of music can result in many matching songs being returned for a given target song. For example, 1 million songs in a database with an average of 0.1% of the song pairs matching results in, on average, 500 matches per song. Most users are only interested in 5 to 10 matches per song and so some techniques for choosing those matches is needed. The methods described use filters derived from the extracted parameters, combined with meta data from other sources.

3.0 BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates how the current invention is used to find music that humans perceive as sounding alike using weighted parameters.

Figure 2 illustrates how the current invention is used to find music that humans perceive as sounding alike using weighted descriptors.

Figure 3 illustrates a song space with clusters of songs with high likeness.

Figure 4 illustrates a method for creating and searching a database.

Figure 5 is an example of an interface used to interact with a database.

4.0 DETAILED DESCRIPTION

4.1 Modeling likeness with parameters (Figure 1)

The invention is illustrated in Figure 1. A database of stored music 101 is played in pairs to one or more humans, step 102, who rate the music on likeness. In the preferred method, the same music is fed into a parameter extractor 103, that uses methods known in the art to extract parameters 104 that are relevant to the perception of music, such as tempo, rhythm complexity, rhythm strength, dynamic range, and harmonicity. Numerous different methods for extracting each of these parameters are known in the art. The model is refined, step 108, by minimizing the difference, calculated in step 106, between the human-derived likeness values and the model-derived values.

An alternative method uses humans to extract parameters that are relevant to the perception of music, such as tempo, rhythm complexity, rhythm strength, singer contribution, number of instruments, type of instruments, and voice timbre. This replaces the machine parameter extractor 103. The model is refined, step 108, by minimizing the difference, calculated in step 106, between the human-derived likeness values and the model-derived values.

4.1.1Collecting human data

One or more humans listen to pairs of songs and judge their similarity on a scale, for example from 1 to 9, where 1 is very dissimilar, and 9 is very similar, step 102.

The objective of using human listeners is to create a model of their perceptual process of "likeness", using a subset of all music, and to apply that model to the set of all music.

The preferred method for collecting human data is to use a panel of ear-pickers who listen to pairs of songs and score their similarity, using a Lickert scale, which is well known in the art.

Another method is to have people visit a web site on which they can listen to pairs of songs and make similarity judgments. These judgments could be on a scale of 1 to 9, or could be yes/no. It is possible to estimate a scalar value of similarity based on a large number of binary judgments using statistical techniques known in the art.

4.1.2 Creating the likeness model

The objective of the model is to predict these perceived differences using the extracted parameters of music. To build the model, a list of numbers is calculated for the comparison of each song to each other song. The list of numbers consists of a value for each parameter where the value is the difference between the parameter value for a first song and the value of the same parameter for a second song. When the model is used to compare one song to another for likeness, the list of parameter differences between the two songs is calculated and these differences are the inputs to the model. The model then yields a number that predicts the likeness that people would judge for the same two songs. The model processes the list of difference values by applying weights and heuristics to each value.

The preferred method of creating a model of likeness is to sum the weighted parameter values. For example, for three parameters, A, B and C, the following steps are used for songs 1 and 2:

STEP 1 – subtract the parameters of each song

$$A_1 - A_2,\ \ B_1 - B_2,\ \ C_1 - C_2$$

STEP 2 – calculate the absolute differences. Our preferred method uses a value of n = 2, but other values can be used.

$$n\sqrt{}(A_1 - A_2)^n,\ \ n\sqrt{}(B_1 - B_2)^n,\ \ n\sqrt{}(C_1 - C_2)^n$$

STEP 3 – weight and sum the differences. The values of the weights (β_1 - β_n) are determined by linear regression, as explained below.

$$\text{Likeness} = \beta_1{*}A_{\text{difference}} + \beta_2{*}B_{\text{difference}} + \beta_3{*}C_{\text{difference}}$$

The value of the weights are determined by a process of linear regression, step 108, which seeks to minimize the difference, step 106, between the human-derived likeness values, step 102, and the output from the model, 105. The preferred model of likeness is:

Likeness = β_0 + β_1*mean loudness + β_2*rhythm strength + β_3*tempo + β_4*dynamic range + β_5* mean harmonicity + β_6* rhythm complexity + β_7* standard deviation harmonicity

Where

$\beta_0 = 0$

$\beta_1 = -0.108$

$\beta_2 = -0.225$

$\beta_3 = -0.127$

$\beta_4 = -0.015$

$\beta_5 = -0.223$

$\beta_6 = -0.122$

$\beta_7 = -0.074$

Another method for deriving likeness is to calculate the correlation coefficients (r) of the parameter values between each pair of songs in the database, and to create a matrix of similarity for the songs, with high correlation equating to high similarity. The parameter values are normalized to ensure that they all have the same range. Song 1 provides the parameters for the x values, and song 2 provides the parameters for the y values in the following formula:

$$r = \Sigma(x_i - x_{mean})\ (y_i - y_{mean})\ /\ \sqrt{}\{[\Sigma\ (x_i - x_{mean})^2][\ \Sigma\ (y_i - y_{mean})^2]\}$$

Where x_{mean} and y_{mean} are the means of the normalized parameter values for songs 1 and 2.

The rationale behind using correlation coefficients is that if the parameters of two songs have a high positive correlation which is statistically significant then the two songs will be judged to be alike.

Yet another method for deriving likeness is to use a neural net to develop the model of likeness 105. In this method, a portion of the human data and parameter data are used to train a network and another portion of the data are used to test the network. One such embodiment, using the neural net toolbox in Matlab™,is described below in steps 1 to 7.

STEP 1) Construct a vector of parameter differences and correlations.

Call this **loaMat**

STEP 2) Normalize the first 69 entries (the difference entries) using the mean

(Norm1Means) and standard deviation (Norm1Stds) in Norm1Stats.mat

```
Norm1loaMat = (loaMat(1:69)-Norm1Means)./Norm1Stds;
```

STEP 3) Divide parameters into classes. The indices corresponding to the classes of different parameter classes for the vector in step 2 are defined by the following:

```
Class{1} = [1:14 72:92]; % Parameter 1
Class{2} = [18:19 70]; % Parameter 2
Class{3} = [71]; % Parameter 3
Class{4} = [19:26]; % Parameter 4
Class{5} = [67:69]; % Parameter 5
```

STEP 4) Apply Normalization and results from Principle Components Analysis (PCA).

a) Normalize each vector created in step 3 using the mean (Norm2Means) and standard deviations (Norm2Stds) in Norm2Stats.mat. Norm2Means and Norm2Stds are 1x5 cell arrays, where each cell corresponds to each of the 5 parameter Classes. Thus, normalize each parameter class by its respective mean and std vectors.

```
for i=1:5
          Norm2loaMatClass{i} = (Norm1loaMat(Class{i}-
Norm2Means{i})./Norm2Stds{i};
end
```

b) To apply the PCA, multiply each of the parameter Class vectors created in step 4a by the corresponding matrix in the PCAMats (1x5) cell array. (i.e., NewClass1 = PCAMats{1}*OldClass1)

```
for i=1:5
       PCAloaMatClass{i} = PCAMats{i}*Norm2loaMatClass{i};
end
```

c) Concatenate Classes into a single vector

```
NetInputVect = [ ];
for i=1:5
   NetInputVect = [NetInputVect; PCAloaMatClass{i}];
end
```

STEP 5) Take the log of some of the parameters. A number of the transformed parameters in NetInputVect were found to be distributed asymmetrically (i.e. Skewed), so take the log. After taking the log, normalize according to Norm2Means and Norm3Stds in Norm3Stats.mat

```
Skewed = [25 27 30 31 32 65 72];
Mins = [-1.393 -2.2379 0 -2.5009 -1.9474 -2.768 -1.5025];
NetInputVect(Skewed) = log(NetInputVect(Skewed)-Mins)+1);
NetInputVect(Skewed) = (NetInputVect(Skewed)-
Norm3Means)./Norm3Stds;
```

STEP 6) Apply results of Clustering. This was done using 2 classes. The estimated class center points in the 74 dimensional space are given in ClassPoints.mat. A 74x2 matrix class ClassPoints is found in ClassPoints.mat. To get the class of a particular song pair, simply compute the euclidian distance between NetInputVect and each of the 2 74x1 vectors in ClassPoints and choose the class associated with the smallest distance.

```
Dist1 = dist(NetInputVect,ClassPoints(:,1));
Dist2 = dist(NetInputVect,ClassPoints(:,2));
if Dist1<Dist2
  Class=1;
else
  Class=2;
end
```

STEP 7) Compute Distance. Two structurally identical networks were trained on each of the two distance classes. To implement the network, do the following

SimNetworkWeights.mat contains the two network objects: net1 net2 as well as the parameters of the output transformNLParameters

```
load SimNetworkWeights.mat
```

Choose the Network

```
if Class==1
   net = net1;
else
   net = net2;
end
```

Compute Input Layer Activations

```
for i=1:5
   InputLayerAct{i} = net.IW{i,i}*NetInputVect(Class{i});
end
```

Compute Hidden Layer Activations

```
        for i=1:5
            HiddenLayerAct = HiddenLayerAct+
        net.LW{9,i}*InputLayerAct{i};
        end
Compute Final Distance Value
        Distance = net.LW{10,9}*HiddenLayerAct;
Apply Saturated Linear Activation Function
        If Distance>1
            Distance = 1;
        elseif Distance<0;
            Distance = 0;
        end
Apply Best Fitting Transform
        p=NLParameters;
        temp = 1./(1+exp(-(p(1)*Distance.^p(6) + p(2))));
        FinalDistance = p(3)*temp.^p(5) + p(4);
```

A fourth method of deriving likeness it to use genre-specific filtering. In this method, it is acknowledged that people may judge two songs as being more or less similar depending on whether or not the songs belong to the same genre. For example, two songs may have very similar parameters, but one may have subtle differences in instrumentation or style that puts it into the genre "country" whereas the other is in the genre "rock." Because the parameters of the songs differ only slightly, a general purpose model may judge them to be similar. However the perceived similarity is low.

To account for this, the songs are identified as belonging to one or more genres and a genre compatibility matrix is used to determine which genres can be presented as matches to other genres. The genre compatibility matrix is derived by asking people to rate the compatibility of pairs of genres, or by a Song Domain Expert. The songs are placed into genres using a combination of the genre information imposed by the song vendor, and by the clustering techniques described below.

A fifth method of deriving likeness acknowledges the context-dependence of likeness comparisons by creating genre-specific models of likeness. For example, if a person listens to a Jazz song, a Punk song, a Classical song, and a Death Metal song, the Punk song and the Death Metal song may be judged to sound alike. However the same two songs may be judged as being dissimilar if the Punk song is presented only with other Death Metal songs. The steps for creating a genre-specific model are the same as for creating a general purpose model, except that the pairs of songs, 101 given to human listeners, 102, are from the same genre. The songs are placed into genres using a combination of the genre information imposed by the song vendor, and by the clustering techniques described below.

4.1.3 Building the likeness data

The preferred method of storing and organizing the parameter differences data is as a multi dimensional vector in a multi dimensional database 107. The resulting matrix contains n*(n-1)/2 cells where n is the number of songs. The model is used by starting with a target song 109, calculating the difference in value for each parameter between the comparison song and the songs in the database, steps 103 - 105, and then applying the model to the difference values to arrive at a value which represents the likeness between the target song and each song in the database.

An alternative method precomputes the 2 dimensional similarity matrix such that each song is connected with only those songs with which there is a match above a predetermined value. Thus, the low matching songs are culled from the similarity matrix. This decreases the size of the database and can increase the search speed.

Yet another method precomputes the 2 dimensional similarity matrix such that each song is connected with only the y closest neighbors, where y is a value chosen to optimize the processing speed and user satisfaction. The preferred value for y is 200.

4.1.4 Limitations of the model

The limit on the goodness of the fit of the model (its predictive power) is determined by, among other things:

(1) The variability between the human responses, step 102. High variability means that people do not agree on the similarities and any model will have poor predictive power.

(2) The intra song variability in the set of all music. High variablity within a song means than any one part of the song will be a poor representation of any other part of the same song. This will impede the listeners' task of judging similarity between such songs, resulting in less accurate data and a less accurate model.

(3) How well the subset represents the set. The ability to create a model depends on the existence of patterns of parameters in similar songs. If the subset on which the model is based does not represent the patterns that are present in the set, then the model will be incorrect.

We have found that a group of 12 human observers had a correlation coefficient of 0.5 or greater in what they consider sounds alike. This indicates that there is sufficient inter-rater reliability to be able to model the process. We can further improve our chances of successfully using the model to predict what sounds alike by:

(a) Only applying the technique to music which has low intra-song variability (e.g. some Classical music has high variability within songs).

(b) Using statistical sampling techniques known in the art (for example, for political polling) to ensure that the subset represents the set.

4.2 Modeling likeness with descriptors (Figure 2)

Figure 2 illustrates an alternative method for finding music that sounds to human listeners like any musical composition selected by a user. Instead of using the parameters directly, they are processed as described in US patent application number 09/556,086 by the same inventors to create a set of descriptors 206. These are used to create a model of likeness 207 by a process similar to that used to create the model of likeness using parameters 105.

4.2.1 Collecting human data

One or more humans listen to pairs of songs and judge their similarity on a scale, for example from 1 to 9, where 1 is very dissimilar, and 9 is very similar, step 202. The objective of using human listeners is to create a model of their perceptual process of "likeness", using a subset of all music, and to apply that model to the set of all music. The preferred or alternative methods of collecting human data described with Figure 1 are used.

4.2.2 Creating the likeness model

The objective of the model is to predict the perceived differences using the modelled descriptors of music. To build the model, a list of numbers is calculated for the comparison of each song to each other song. The list of numbers consists of a value for each descriptor where the value is the difference between the descriptor value for a first song and the value of the same descriptor for a second song. When the model is used to compare one song to another for likeness, the list of descriptor differences between the two songs is calculated and these differences are the inputs to the model. The model then yields a number that predicts the likeness that people would judge for the same two songs. The model processes the list of descriptor difference values by applying weights and heuristics to each value.

The preferred method of creating a model of likeness is to sum the weighted descriptor difference values. For example, for three descriptors, A, B and C, the following steps are used for songs 1 and 2:

STEP 1 – subtract the descriptors of each song

$A_1 - A_2$, $B_1 - B_2$, $C_1 - C_2$

STEP 2 – calculate the absolute differences. Our preferred method uses a value of n = 2, but other values can be used.

$n\sqrt{(A_1 - A_2)^n}$, $n\sqrt{(B_1 - B_2)^n}$, $n\sqrt{(C_1 - C_2)^n}$

STEP 3 – weight and sum the differences. The values of the weights (β_1 - β_n) are determined by linear regression, as explained below.

$$\text{Likeness} = \beta_1{*}A_{\text{difference}} + \beta_2{*}B_{\text{difference}} + \beta_3{*}C_{\text{difference}}$$

The value of the weights are determined by a process of linear regression, step 210, which seeks to minimize the difference, step 208, between the human-derived likeness values, step 202, and the output from the model. The preferred model of likeness using the descriptors Energy, Tempo, Happiness, Anger, and Danceability is:

$$\text{Likeness} = \beta_0 + \beta_1{*}\text{Energy} + \beta_2{*}\text{Tempo} + \beta_3{*}\text{Happiness} + \beta_4{*}\text{Anger} + \beta_5{*}\text{Danceability}$$

$\beta_0 = 0$

$\beta_1 = 11.4$

$\beta_2 = 0.87$

$\beta_3 = 4.1$

$\beta_4 = 6.3$

$\beta_5 = 15.94$

4.2.3 Organizing and storing the likeness data

The preferred and alternative methods of organizing and storing the parameter differences data for calculating likeness are also used when the process uses descriptors. In addition, there is yet another alternative for calculating likeness. It involves precomputing a series of hierarchical listings of the songs organized by their descriptor values. For example, all of the songs in the database are organized into nine classes according to their energy values. Then all of the songs with an energy value of 9 are organized by their tempo values, then all of the songs with energy value 8 are organized into their tempo values, and so on, through all levels of all of the descriptors. The sorting results in a maximum of L^n similarity classes, where L is the number of levels of each descriptor and n is the number of descriptors. In this case, 59049 classes. The songs in each class have the identical descriptor values. This provides an alternative type of likeness.

4.3. Computing the database

The number of combinations of songs increases with the square of the number of songs. For large numbers of songs this creates computational challenges because of :

(a) The amount of data that must be read from the database of parameters 104,

(b) The number of comparisons that have to be performed, step 105, and

(c) The amount of data that must be written to the database of parameter differences, 107.

Methods for dealing with these challenges are described below.

4.3.1 Precomputing Partitions in the Database to Decrease the Number of Comparisons (Figure 3)

Clustering is the grouping of objects into naturally present categories, or partitions. One way to do this is to compute a likeness value (or distance) between objects. This likeness space is then transformed into naturally present categories, or partitions.

Figure 3 depicts a song space where dots, 301, represent individual songs. The songs are grouped or "clustered" according to their perceived likeness. Pairs of songs within a circle, 302, are judged to be more alike than pairs of songs outside of a circle, 301 and 303. Figure 3 also depicts a hierarchy of clusters; the root cluster being the overall song space;304, within which are 4 clusters

Since all songs within a partition have high likeness, and those in separate partitions have low likeness, then likeness need only be computed for songs within a partition. Assuming that the number of songs in each partition is evenly distributed, this allows us to reduce the number of likeness computations, N, by N/k (k = number of partitions). Songs can be placed into the correct clusters by choosing representative songs (the "leader" songs) from each cluster and comparing every song with all leader songs. Each song is placed in the cluster represented by the leader song to which it is most alike.

Two clustering structures known in the art are hierarchical and non-hierarchical. In hierarchical clustering, every object in the set of all objects fits into one or more clusters, and any two clusters are either disjoint or one cluster includes the other. In non-hierarchical clustering, every object in the set of all objects fits into exactly one cluster, and any one cluster contains one or more objects.

A PISA, or Partition Intrinsic Subjective Attribute is a measure of how well a song fits into a category of songs. The steps for creating a PISA are as follows:

Step 1: Select a representative subset of songs from the universe of all music. The preferred method is to use a random sampling process to select 5,000 songs.

Step 2: Calculate a likeness matrix on the representative subset of songs. The preferred method uses the likeness model described earlier to compare all 12,497,500 pairs.

Step 3: Cluster the likeness data. The preferred method uses a hierarchical clustering structure.

Step 4: The clustering results are reviewed by a song domain expert (SDE) who defines partitions by identifying a leader song (or songs) and setting a minimum likeness threshold above which a song must rise when compared to the leader(s) before it can be a member of a PISA cluster. The preferred method for defining partitions uses the following criteria:

- Partitions are distinct from one another -- i.e. sufficiently far from one another in similarity space for the SDE to recognize them as genres.
- All songs within the partition sound mostly alike. (by "mostly" we mean with less than a 5% false positive error rate).
- Partitions are generalizable -- i.e. similarity between a partition and a given song in a large song space, given the song falls above an acceptability threshold, will sound like all songs in the representative set.

Step 5: PISA values are calculated on any larger set of songs selected for the database.

A given PISA value is defined as follows:

ISAVal = 1/N*(sum (sim (i, k)) where k is the identifier for each song defining a partition and N is the total number of songs that define that partition.

For example, if a partition is defined by songIds = [1,2,3], then

ISAVal = 1/3*(sim(i,1) + sim(i,2) + sim(i,3))

Threshold values for each partition are used on each song to determine whether the song is a member of each defined partition. Partition membership, as well as the song's PISA value is stored for each song.

Another method of clustering uses unsupervised clustering techniques known in the art, such as the Kohonen learning rule (Kohonen, T. (1995). *Self-Organizing Maps* Springer-Verlag, Berlin-Heidelberg-New York). In this method the steps used are as follows.

Step 1: Select a representative subset of songs from the universe of all music. The preferred method is to use a random sampling process to select 256 songs.

Step 2: Use a self organizing neural net to arrange the songs into clusters on the basis of their parameter values.

Step 3: The clustering results are reviewed by a song domain expert (SDE) who defines legitimate clusters by listening to the songs.

Step 4: The SDE then identifies a leader song (or songs) in each cluster.

Step 5: The leader songs are used in the same way as the leader songs described earlier in the PISA method.

4.3.2 Decreasing the Amount of Data Read from and Written to the Database

The time that it takes to read and write data from the database is a limiting factor to the speed of the computation of likeness. Decreasing the number of reads and writes improves performance.

The preferred method for decreasing the amount of data read from the database is to decrease the amount of information contained within the parameters by decreasing the precision of the numeric values.

Another method loads all of the parameter data and the likeness models into RAM and then calculates and stores all of the likeness values without further recourse to the database. Selected likeness values are then written to the database.

4.4 Interacting with the database (figures 4 and 5)

4.4.1 Methods for Presenting Likeness Data

Once the database has been created, it can be combined with other meta data about the music, such as the name of the artist, the name of the song, the date of recording etc. One method for searching the database is illustrated in Figure 4 which is based on the methods using descriptors shown in Figure 2. A large set of music 401, representing "all music" is sent to the parameter extractor 402. The descriptors are derived and then combined with other meta data and pointers to the location of the music, for example URLs of the places where the music can be purchased, to create a database 405. A user can interrogate the database 405, and receive the results of that interrogation using an interface 406.

An example of an interface is illustrated in Figure 5. A user can type in textual queries using the text box 504. For example; "Song title: Samba pa ti, Artist: Santana." The user then submits the query by pressing the sort by similarity button 503. The song "Samba pa ti" becomes the target song, and appears in the target box 501. The computer searches the n dimensional database 405 of vectors made up of the song descriptors, looking for the smallest vector distances between the target song and other songs. These songs are arranged in a hit list box 502, arranged in increasing vector distance (decreasing similarity). An indication of the vector distance between each hit song and the target song is shown beside each hit song, expressed as a percent, with 100% being the same song. For example, the top of the list may be "Girl from Ipanema by Stan Getz, 85%."

Another type of query allows a user to arrange the songs by profile. First, the user presses the sort by number button 509 which lists all of the songs in the database in numerical order. The user can scroll through the songs using a scroll bar 506. They can select a song by clicking on it, and play the song by double clicking on it. Any selected song is profiled by the slider bars 505. These show the scalar values of each of several descriptors. The current preferred method uses energy, danceability and anger. Pressing the sort by profile button 510 places the highlighted song in the target box 501 and lists songs in the hit box 503 that have the closest values to the values of the target song.

Yet another type of query allows the user to sort by similarity plus profile. First a target song is chosen and the songs in the hit box 502 are listed by similarity. Then the user performs a search of this subset of songs by using the slider bars 505. For example, the slider values of the original target song are 5,6,7 for energy, danceabilty and anger respectively. The user increases the energy value from 5 to 9 and presses the search similar songs button 507. The profile of the target song remains as a ghosted image 508 on the energy slider. The computer searches the subset of songs for a song with values of 9, 6 and 7. Songs with that profile are arranged in decreasing order of similarity. Songs without that profile are appended, arranged by decreasing similarity. The target song remains in the target box 501. A user can choose a new target song by clicking on a song in the hit list box 502.

The likeness database may be quickly searched by starting with any song and immediately finding the closest songs. A more enhanced search combines a similarity search with a descriptor search and the descriptor adjustment step described above. The steps are:

1a. Find a list of likeness matches, including some that are somewhat different

1b. Present only the acceptable likeness songs.

When a search adjusted by a scalar descriptor is requested, rank the entire list of likeness matches (1a) by the descriptor to be adjusted.

Present a new list based on the adjusted values with the best matches at the top. This means that the likeness list (1a.) compiled for the original search is much broader than the list displayed for the user(1b), and includes songs that are less similar to the initial target song than would be tolerated by the listener. These poorer likeness matches lie below the presentability threshold for the initial target.

4.4.1 Filtering Likeness to Improve Performance

The perceived performance of the database can be enhanced by filtering the results of likeness searches.

The preferred method uses genres and allows a user to instruct the system to return search results from the same, or different genres than the target song. The genres are derived from a combination of the clustering methods described earlier, or from information imposed by vendors.

A second method uses meta data for individual songs to construct song "parents." A song parent is a grouping of songs with some common elements. For example, the songs of an artist, album or genre may have some common elements and are parents of individual songs. The preferred method of constructing parents is to use a grouping algorithm. One way in which parents can be used is to create a compatibility matrix of parents, and use that to ensure that songs from incompatible parents are not presented as matches to target songs.

A third method uses knowledge about the popularity of songs as a filter. For example, if a target song results in 200 likeness matches, the results may be presented in order of decreasing popularity. Popularity is measured by several methods, including sales volume, listening activity, and search activity.

A fourth method uses user preference to create a taste template. A taste template describes the ranges of the parameter values for a set of songs that a user likes. The taste template can be used to filter the output of the likeness model to better match an individual's tastes. The taste templates can also be compared across multiple users using collaborative filtering. The result of collaborative filtering uncovers taste preferences between people and parameters. For example, people who like songs with meta data or descriptors x, y, z also like songs with meta data or descriptors u,v,w. Therefore, kif someone is searching for x,y,z it is likely that they would like to be presented with songs that contain u,v,w.

A fifth method uses the likeness model to identify songs that have a large number of high likeness matches, and presents those in preference to songs that have few matches or low likeness matches. The former songs are said to have a high neighborhood density. Neighborhood density is a measure of the likelihood a given song will have good likeness matches. More explicitly, Neighborhood density is a weighted measure of how many songs appear as matches to a target song within a specified likeness distance. An example of the use of neighborhood density is when a user searches the database for songs in the genre "soundtrack" with the mood "aggressive." The songs that are returned can be chosen by a number of criteria, including alphabetical order of title, most recently recorded, and neighborhood density. The use of neighborhood density increases the chances that the presented songs will have high likeness matches.

Constraining the User to Use only Precomputed Target Songs

As described above, the system is used to find songs within a database that are like a target song. In the most generic implementation, a user can load a database of songs, compute the parameters for the songs, input a target song, compute the parameters for the target, and compute the likeness between the target and each song in the database using the system's model of likeness based on human perception of similarity. However, for a database of any substantial size, the computation of parameters for the target song and computation of likeness to other songs in the database requires so much time and processor dedication as to be impractical for large scale implementation.

Part of the solution to this efficiency problem, as described above, is to precompute the parameters for each song in the database. Another part of the solution, as described above, is to precompute leader songs or partitions between songs in the database. With these precomputations, if the user provides a target song, the only computation remaining is to compute parameters for the target and compute comparisons to the partitions in the database. However, additional speed enhancements are required to make the system practical.

The final required enhancement, which makes a crucial difference, is to constrain the user's choice of target songs to the existing songs in the database and not allow the user to input a recording of a song that is not already in the database. This means that the computation of parameters for the target song has already been done and the computation of comparisons to other songs or partitions in the database has already been done. The problem is thereby simplified for a search in a very large database.

What is claimed is:

1. A method for allowing a user to find musical recordings like a selected musical recording, comprising:

   a) assembling a database of records, each record associating a musical recording, a title for the musical recording, and a set of precomputed measures of likeness between the musical recording and other musical recordings having records within the database;

   b) receiving from a user a selection of one of said titles within said database;

   c) using said set of precomputed likeness measures to find within the database records having associated musical recordings that are similar to the musical recording associated with said selected title;

   d) presenting to the user a list of titles from the database, each title associated with a record found using said precomputed set of likeness measures.

2. The method of claim 1 where the set of precomputed measures of likeness are generated by extracting parameters from each musical recording and computing comparisons between parameters for pairs of musical recordings.

3. The method of claim 1 where the method used to compute likeness measures between musical recordings was empirically developed based on perceptions reported by one or more human listeners.

4. The method of claim 1 where the method used to compute likeness measures was developed using a neural network.

5. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 1.

6. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 2.

7. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 3.

8. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 4.

9. A method for using precomputed parameters for musical recordings to allow users to find similar musical recordings, comprising:

   a) for each of a plurality of musical recordings, computing a plurality of parameters extracted from the contents of each musical recording;

b) receiving from a user a specification of a target musical recording for which parameters were precomputed and a specification of a set of other musical recordings for which parameters were precomputed; and

c) using said precomputed parameters, computing likeness measures between said specified target musical recording and said specified set of other musical recordings; and

d) presenting to the user a specification of one or more musical recordings from the specified set of other musical recordings having relatively high computed likeness measures to the specified target recording.

10 The method of claim 9 where the method used to compute likeness measures between musical recordings was empirically developed based on perceptions reported by one or more human listeners.

11. The method of claim 9 where the method used to compute likeness measures was developed using a neural network.

12. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 9.

13. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 10.

14. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 11.

15. A method for building a computational model of human perception of likeness between musical recordings, comprising:

a) extracting from each of at least 5 electronic representations of musical recordings at least two numeric parameters;

b) receiving from one or more human listeners who compare pairs of the musical recordings an indication of the human's perception of likeness for each compared pair of recordings;

c) for each compared pair of the recordings, comparing each numeric parameter of one recording in the pair with the corresponding parameter of the second recording in the pair using an algorithm which produces a parameter comparison number;

d) for each compared pair of the recordings, combining the parameter comparison numbers with a weighting for each parameter comparison number to compute a single difference number representing the difference between the two recordings of the pair;

e) adjusting the weightings for the comparison numbers to find a set of weightings where each computed difference number for each pair of recordings most closely matches perceptions reported for the pair of recordings by the one or more human listeners.

16. The method of claim 15 where the algorithm includes subtraction of parameter values.

17. The method of claim 15 where the algorithm includes computing a correlation between parameter values.

18. The method of claim 15 where the algorithm employs a neural network.

19. The method of claim 15 where, prior to the step of comparing the numeric parameters:

a) the parameters for each recording are combined with a weighting for each parameter to compute a single number representing a descriptor for that recording, where

b) the weightings were previously determined by adjusting the weightings to find a set of weightings where each computed descriptor for each recording most closely matches perceptions reported for the recording by one or more human listeners, and

c) the descriptors are then used in the step of comparing the numeric parameters in place of the parameters.

20. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 15.

21. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 16.

22. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 17.

23. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 18.

24. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 19.

25. A method for creating a partitioned database of identifiers for musical recordings, comprising:

a) selecting a set of musical recordings that is representative of additional musical recordings;

b) extracting from the contents of each musical recording of the set of musical recordings and the additional musical recordings at least two numeric parameters;

c) associating an identifier with each recording of the set of musical recordings and the additional musical recordings;

d) assembling the identifiers for the set of musical recordings into a database;

e) computing from the extracted parameters for each of a plurality of pairs of the recordings associated with each identifier in the database a number which represents the difference between the recordings of the pair;

f) based on said computed difference numbers, partitioning the database such that each partition contains identifiers of musical recordings that are similar to each other; and

g) placing identifiers for said additional musical recordings into one or more of said partitions.

26. The method of claim 25 where computation of the number representing the difference between the recordings of the pair is done with a model based on human perception.

27. The method of claim 25 where computation of the number representing the difference between the recordings of the pair is done based on distribution of parameter values.

28. The method of claim 25 further comprising, within each partition, selecting one or more representative musical recordings to represent the partition and placing identifiers for additional musical recordings into partitions by comparing parameters for the recording associated with the identifier to parameters for the one or more representative musical recordings.

29. The method of claim 28 where the comparison step includes subtraction of

parameter values.

30. The method of claim 28 where the comparison step includes computing a correlation between parameter values.

31. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 25.

32. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 26.

33. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 27.

34. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 28.

35. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 29.

36. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 30.

37. A method for searching a database of data records associated with musical recordings to find a desired recording, comprising:

a) identifying a comparison data record associated with a musical recording in a computer readable database containing a plurality of data records, each associated with a musical recording, the data records each comprising two or more scalar descriptors, each descriptor numerically describing the recording of musical with which the data record is associated, where each descriptor was generated by:

1) extracting from an electronic representation of the recording of musical at least two numeric parameters;

2) combining the numeric parameters with a weighting for each parameter to compute a single number representing the descriptor for that recording, where the weightings were previously determined by:

3) extracting from an electronic representation of each of at least 5 musical recordings the same at least two numeric parameters;

4) for each recording, combining the numeric parameters with a weighting for each parameter to compute a single number

representing the descriptor for that recording;

5) adjusting the weightings for the parameters to find a set of weightings where each computed descriptor for each recording most closely matches perceptions reported for the recording by one or more human listeners; and

b) searching the database to find data records with descriptors that are similar to the descriptors of the comparison record.

38. The method of claim 37 further including, prior to searching the database, specifying that one of the descriptors of the comparison data record should be adjusted with an increase or a decrease, and the searching step is based on the descriptors of the comparison data record as adjusted.

39. The method of claim 37 where the weightings for the parameters are adjusted using a neural network.

40. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 37.

41. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 38.

42. Computer readable data containing a computer program which causes one or more computers to perform the method of claim 39.

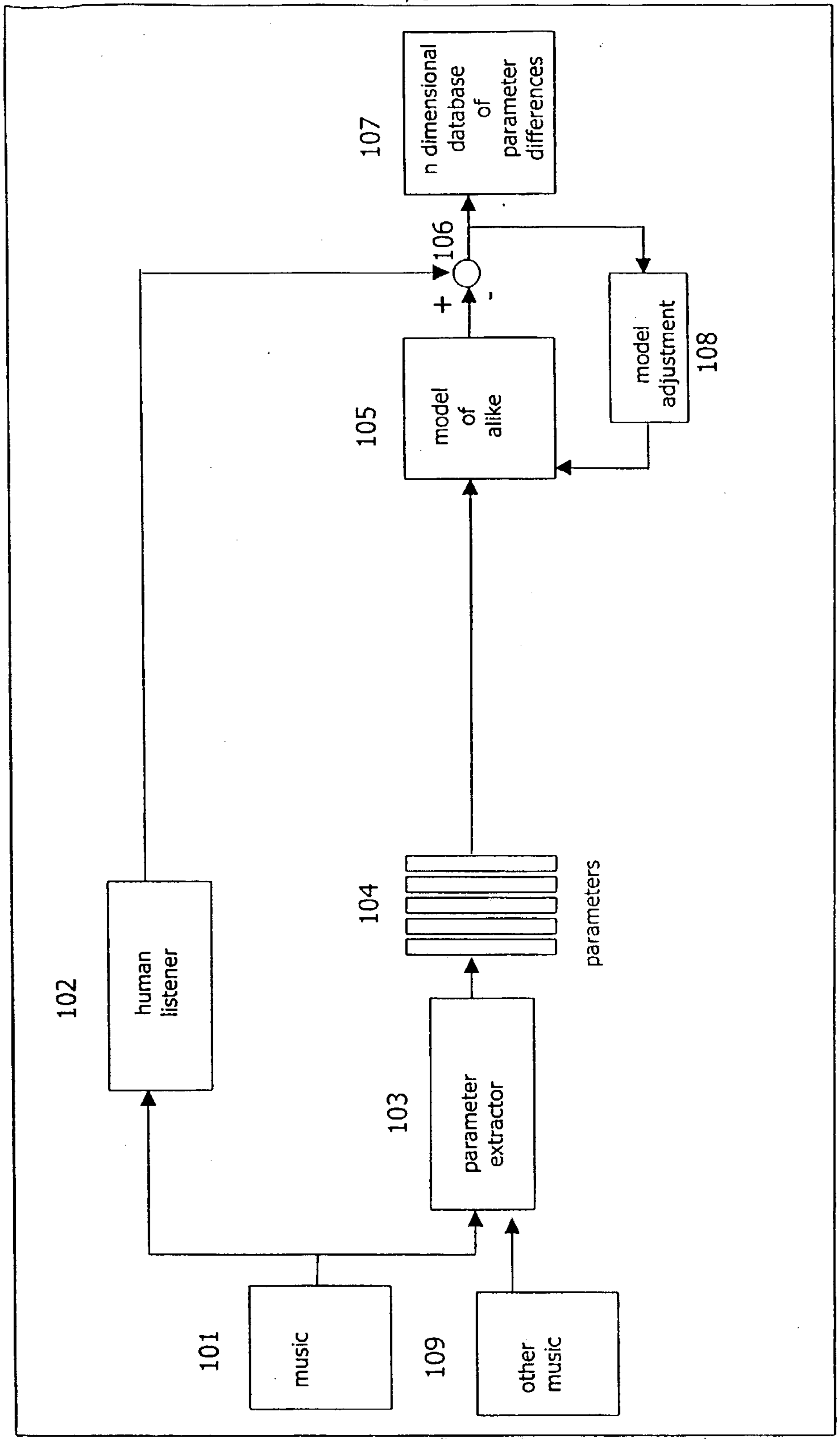

101
music
109
other music
103
parameter extractor
104
parameters
102
human listener
105
model of alike
+
-
106
model adjustment
108
107
n dimensional database of parameter differences

Figure 1

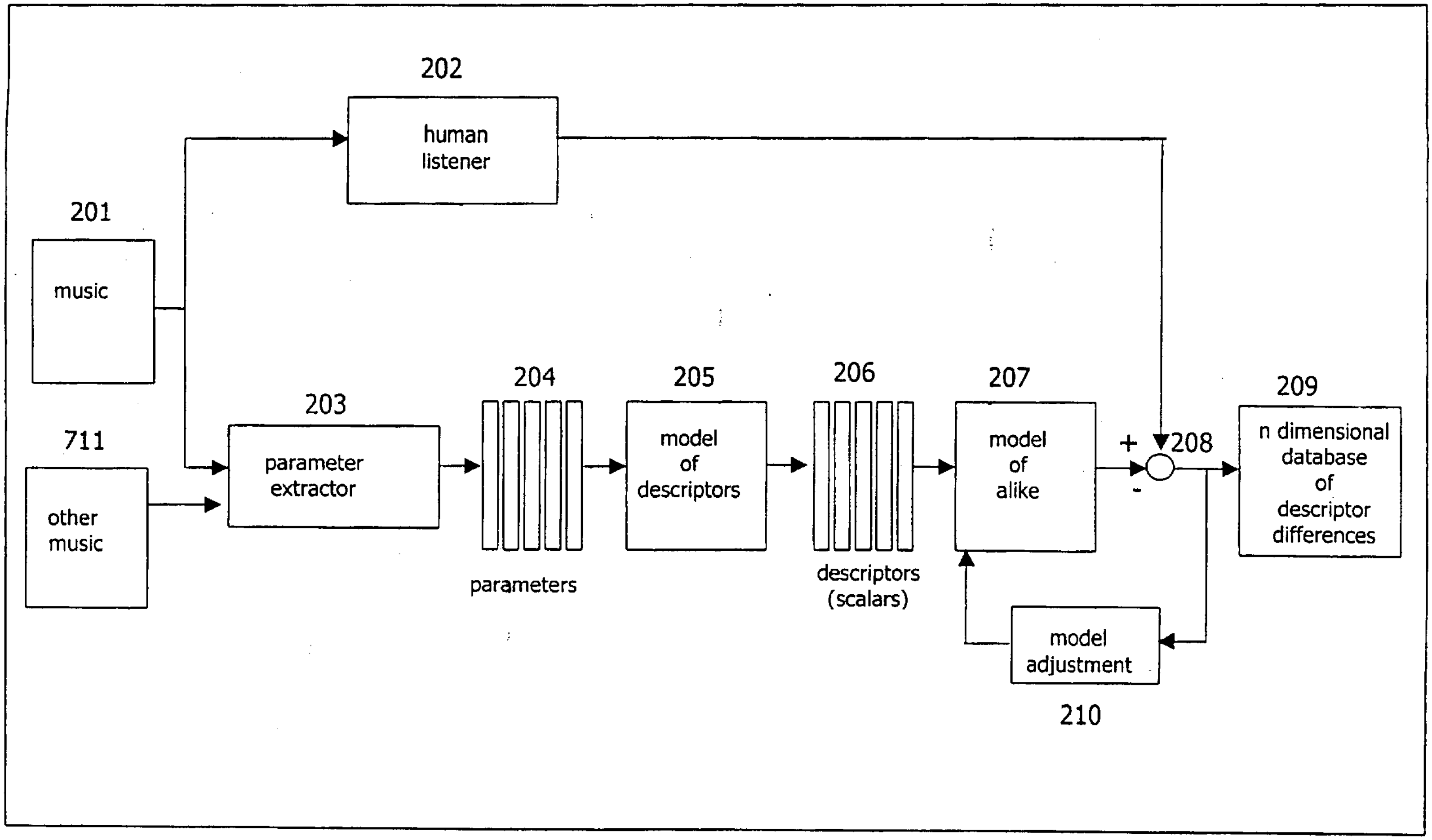

202
human listener
201
music
711
other music
203
parameter extractor
204
parameters
205
model of descriptors
206
descriptors (scalars)
207
model of alike
+
-
208
209
n dimensional database of descriptor differences
model adjustment
210

Figure 2

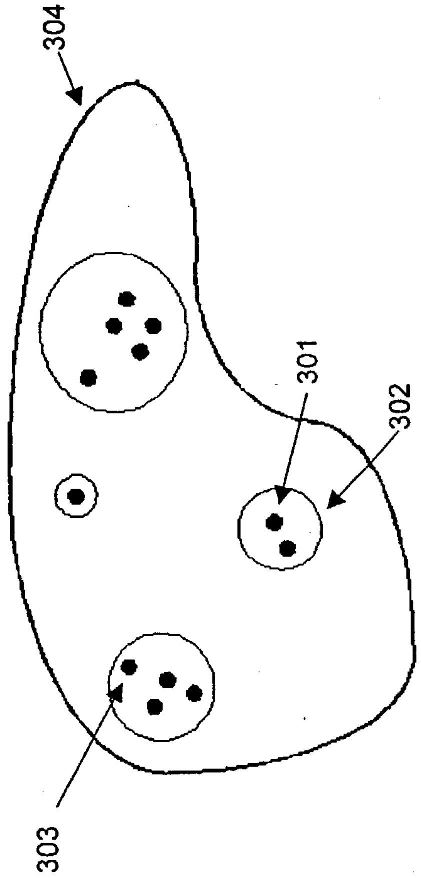


Figure 3

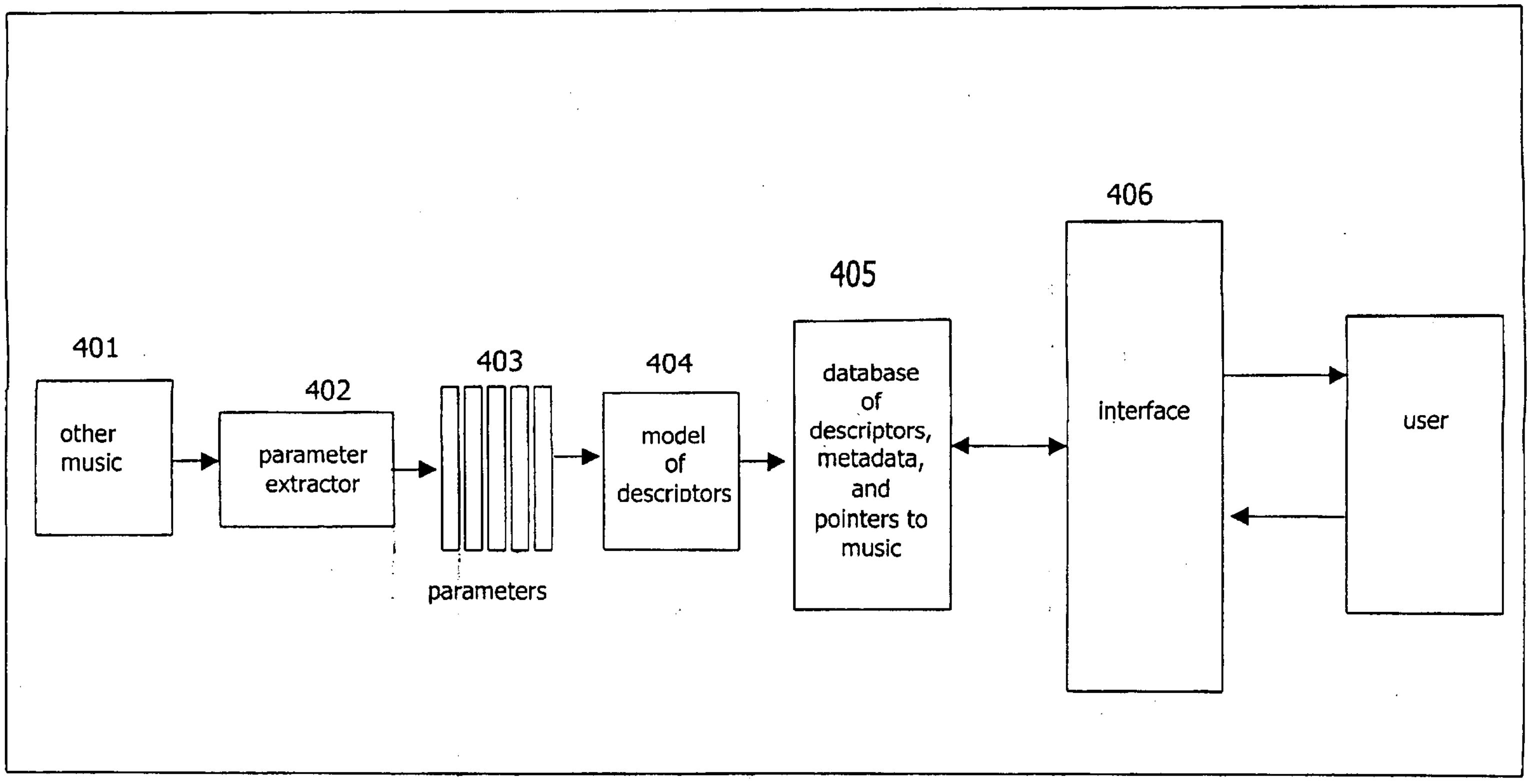

401
other music
402
parameter extractor
403
parameters
404
model of descriptors
405
database of descriptors, metadata, and pointers to music
406
interface
user

Figure 4

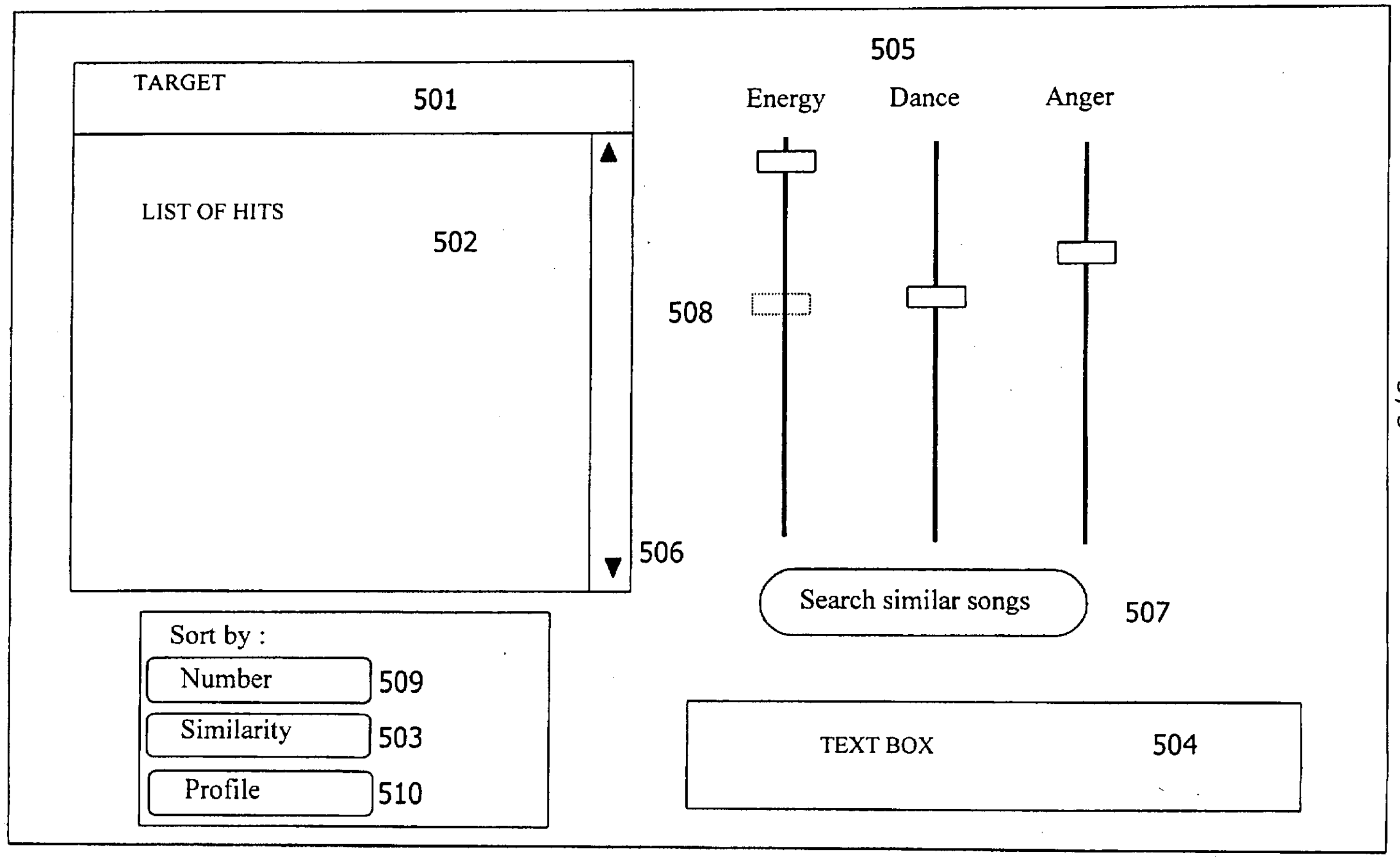

505
TARGET
501
LIST OF HITS
502
506
Energy
Dance
Anger
508
Search similar songs
507
Sort by :
Number
509
Similarity
503
Profile
510
TEXT BOX
504

Figure 5